WE CLAIM:

1. A method of obtaining a resolution-based proof of unsatisfiability using a SAT procedure for a hybrid Boolean constraint problem comprising:

representing constraints as a combination of clauses and interconnected gates; and

obtaining the proof as a combination of clauses, circuit gates and gate connectivity constraints sufficient for unsatisfiability.

2. The method of claim 1, wherein the proof is obtained using a process comprising:

a) stopping if no more decisions are needed;

b) selecting a hitherto unselected decision variable and assigning it a value;

c) performing boolean constraint propagation;

d) going to step a if no conflicts are generated;

e) performing conflict analysis and generating a conflict constraint for learning;

f) recording and associating reasons, consisting of gates and clauses, with the generated conflict constraint;

g) backtracking to step a if backtracking is possible; and

h) recursively marking constraints consisting of gates and clauses, associated as reasons with the final conflict and marked conflict constraints, said marked constraints forming the proof of unsatisfiability.

3. The method of claim 2, wherein boolean simplification is performed prior to preprocessing.

4. The method of claim 3, wherein reasons for the boolean simplification are recorded.

5. The method of claim 1, wherein an association between a learned constraint and its reasons are saved across at least two runs of the SAT procedure .

6. The method of claim 1, wherein heuristics are varied over at least two runs of the SAT procedure.

7. The method of claim 1, wherein prioritization techniques are used to derive a smaller set of marked constraints.

8. A method for determining an abstract model of an original circuit, comprising

a) marking constraints for unsatisfiability at decision level 0;

b) keeping only a subset of the marked constraints in step a, said subset being in a transitive fanin of all marked constraints that are external; and

c) designating a revised circuit formed from the subset identified in step b as the abstract model.

9. The method of claim 8, wherein marking constraints required for unsatisfiability using a sub-process comprising:

i) stopping if no more decisions are needed;

ii) select a hitherto unselected decision variable and assigning it a value;

iii) performing boolean constraint propagation;

iv) going to step a if no conflicts are generated;

v) performing conflict analysis and generating a conflict constraint for learning;

vi) recording and associating reasons, consisting of gates and clauses, with the generated conflict constraint;

vii) backtracking to step a if backtracking is possible; and

viii) recursively marking constraints consisting of gates and clauses, associated as reasons with final conflict and marked conflict constraints.

10. A method for determining an abstract sequential circuit model of an original sequential circuit model, comprising:

a) adding to the original sequential circuit model, interface propagation constraints for each flip-flop to capture equality between an input and output of a flip-flop across successive time frames, and an initial value constraint for each flip-flop;

b) marking constraints required for unsatisfiability of external constraints across multiple cycles of operation of the sequential circuit;

c) forming the abstract sequential circuit by including combinational fanin cones of marked external constraints and flip-flops corresponding to marked interface propagation constraints and flip-flops corresponding to marked initial value constraints.

11. A method for determining an abstract sequential circuit model of an original sequential circuit model, comprising:

a) adding to the original circuit model, interface propagation constraints for each flip-flop to capture equality between an input and output of a flip-flop across successive time frames, and an initial value constraint for each flip-flop;

b) marking constraints required for unsatisfiability of external constraints across multiple cycles of operation of the sequential circuit;

c) forming the abstract sequential circuit by including combinational fanin cones of marked external constraints and flip-flops

corresponding to marked interface propagation constraints only, and by adding constraints for initial values of flip-flops with marked initial value constraint only.

12. A method for determining an abstract sequential circuit model of an original sequential circuit model, comprising:

a) adding to the original sequential circuit model, interface propagation constraints for each flip-flop to capture equality between an input and output of a flip-flop across successive time frames, and an initial value constraint for each flip-flop;

b) marking constraints required for unsatisfiability of external constraints across multiple cycles of operation of the sequential circuit; and

c) forming the abstract sequential circuit by including combinational fanin cones of marked external constraints and flip-flops corresponding to marked interface propagation constraints only, and by adding lazy constraints for initial values of flip-flops with marked initial value constraint only.

13. The method of claim 10 wherein the abstract sequential circuit is obtained based on a fixed number of time-frame unrolling of the sequential circuit.

14. The method of claim 13, wherein the abstract sequential circuit model of the original sequential circuit based on k time-frame unrollings is generated based on an abstract model based on fewer than k time-frame unrollings.

15. The method of claim 14, wherein the number of time-frame unrollings is increased progressively until the abstract model satisfies some criteria.

16. The method of claim 14, wherein the number of time-frame unrollings is increased progressively until the abstract model is stable for a certain number of unrollings.

17. The method of claim 14 wherein the number of time-frame unrollings is increased progressively such that if a SAT formula is satisfiable when the number of time-frame unrollings is k and the abstraction was attempted from the abstract model obtained at the n^{th} time frame ($n < k$), an attempt is repeated to obtain the abstract model at the k^{th} time frame starting from an abstract model obtained at a time frame $m > n$.

18. The method of claim 8, wherein at least one external constraint is applied in a lazy manner.

19. The method of claim 18, wherein the external constraint is replaced with at least one clause that applies the at least one constraint but does not lead to an immediate implication.

20. The method of claim 19, wherein the at least one clause includes a dummy variable.

21. The method of claim 10, wherein at least one constraint corresponding to initial value of a flip-flop is applied in a lazy manner.

22 The method of claim 19, wherein the at least one lazy constraint is replaced with at least one clause that applies the constraint but does not lead to an immediate implication.

23. The method of claim 23, wherein the at least one clause includes a dummy variable.

24. The method of claim 10, wherein at least one external constraint is applied in a lazy manner.

25. The method of claim 24, wherein the external constraint is replaced with at least one clause that applies the at least one constraint but does not lead to an immediate implication.

26. The method of claim 25, wherein the at least one clause includes a dummy variable.

27. A method for determining an abstract model of an original circuit, comprising

a) adding external constraints to the original circuit model based on a need to check temporal properties of the original circuit, and for enforcing environment constraints;

b) marking constraints for unsatisfiability at decision level 0;

c) keeping only a subset of the marked constraints in step a, said subset being in a transitive fanin of all marked constraints that are external; and

d) designating a revised circuit formed from the subset identified in step b as the abstract model.

28. The method of claim 27, wherein marking constraints required for unsatisfiability using a sub-process comprising:

i) stopping if no more decisions are needed;

ii) selecting a hitherto unselected decision variable and assigning it a value;

iii) performing boolean constraint propagation;

iv) going to step i if no conflicts are generated;

v) performing conflict analysis and generating a conflict constraint for learning;

vi) recording and associating reasons, consisting of gates and clauses, with the generated conflict constraint;

vii) backtracking to step a if backtracking is possible; and

viii) recursively marking constraints, consisting of gates and clauses, associated as reasons for final conflict and marked conflict constraints.

29. A method for determining an abstract sequential circuit model of an original sequential circuit model, comprising:

a) adding external constraints to the original circuit model based on a need to check temporal properties of the original circuit and for enforcing environment constraints;

b) adding interface constraints to the original sequential circuit model to capture equality between an input and output of a flip-flop across successive time frames, including its initial value constraint at depth 0;

c) marking constraints for unsatisfiability of external constraints across multiple cycles of operation of the sequential circuit; and

d) forming the abstract sequential circuit by including combinational fanin cones of marked external constraints and flip-flops corresponding to only the marked interface constraints.

30. The method of claim 28 wherein the abstract sequential circuit is obtained based on a fixed number of time-frame unrolling of the sequential circuit, wherein the property is unsatisfiable only for that fixed number of time-frames.

31. The method of claim 28 wherein the abstract sequential circuit is obtained based on a fixed number of time-frame unrolling of the sequential circuit, wherein the property is unsatisfiable for all time frames up to the fixed number of time-frames.

32. The method of claim 30, wherein the abstract sequential circuit model of the original sequential circuit based on k time-frame unrollings is generated based on an abstract model based on fewer than k time-frame unrollings.

33. The method of claim 31, wherein the abstract sequential circuit model of the original sequential circuit based on k time-frame unrollings is generated based on an abstract model based on fewer than k time-frame unrollings.

34. The method of claim 32 wherein the number of time-frame unrollings is increased progressively until the abstract model satisfies some criteria.

35. The method of claim 32, wherein the number of time-frame unrollings is increased progressively until the abstract model is stable for a certain number of unrollings.

36. The method of claim 32 wherein the number of time-frame unrollings is increased progressively such that if a SAT formula is satisfiable when the number of time-frame unrollings is k and the abstraction was attempted from the abstract model obtained at the n^{th} time frame ($n < k$), an attempt is repeated to obtain the abstract model at the k^{th} time frame starting from an abstract model obtained at a time frame $m > n$.

37. An iterative abstraction method for obtaining an abstract model for verifying a design of a circuit, comprising:

a) receiving a concrete design of the circuit;

b) receiving a correctness property;

c) assigning the concrete design as an nth-abstract model , where n =1;

d) performing bounded model checking with resolution-based proof analysis on n-th abstract model;

e) extracting an (n+1)th-abstract model;

f) stopping if further abstraction is not necessary based on a criteria; and

g) assigning n=n+1 and going to step d.

38. The method of claim 37, wherein in d, if counterexample is found while extracting the (n+1)th-abstract model, a new nth-abstract model is generated.

39. The method of claim 37, where if no counterexample is found up to a specified maximum depth, a sufficient set of reasons is chosen heuristically up to the specified maximum depth to generate an accumulated sufficient model which is assigned as the (n+1)th abstract model.

40. The method of claim 39, where the sufficient set of reasons that remain unchanged for a number of time frames is chosen.

41. The method of claim 37, if it is decided to stop further abstraction, following steps are performed:

f1) verifying the (n+1)th abstract model and stopping if a conclusive result is found.

42. The method of claim 37, wherein, if it is decided to stop further abstraction, following steps are performed:

f1) verifying the (n+1)th abstract model and proceeding to step g if an inconclusive result is found.

43. The method of claim 37, wherein, if it is decided to stop further abstraction, following steps are performed:

f1) verifying the (n+1)th abstract model;

f2) handling counterexamples and proceeding to step g if a counterexample is found.

44. The method of claim 41 wherein symbolic model checking is used for verification.

45. The method of claim 42 wherein symbolic model checking is used for verification.

46. The method of claim 43 wherein symbolic model checking is used for verification.

47. The method of claim 41 wherein symbolic traversal technique is used for verification.

48. The method of claim 42 wherein symbolic traversal technique is used for verification.

49. The method of claim 43 wherein symbolic traversal technique is used for verification.

50. The method of claim 43, wherein a counterexample on the nth abstract model is handled using the following:

i) stopping if the current nth abstract model is the concrete design;

ii) running bounded model checking up to a depth greater than the depth of the counterexample on the (n-1)th-abstract model and generating an accumulated abstract model as the new nth abstract model if the bounded model checking can be completed; and

iii) refining the current abstract model to obtain the new nth abstract model if the bounded model checking can not be completed.

51. The method of claim 38, wherein a counterexample on the nth abstract model is handled using the following:

i) stopping if the current nth abstract model is the concrete design;

ii) running bounded model checking up to a depth greater than the depth of the counterexample on the (n-1)th-abstract model and generating an accumulated abstract model as the new nth abstract model if the bounded model checking can be completed; and

iii) refining the current abstract model to obtain the new nth abstract model if the bounded model checking can not be completed.

52. The method of claim 27, wherein at least one external constraint is applied in a lazy manner.

53. The method of claim 52, wherein the external constraint is replaced with at least one clause that applies the at least one constraint but does not lead to an immediate implication.

54. The method of claim 53, wherein the at least one clause includes a dummy variable.

55. The method of claim 29, wherein at least one external constraint is applied in a lazy manner.

56. The method of claim 55, wherein the external constraint is replaced with at least one clause that applies the at least one constraint but does not lead to an immediate implication.

57. The method of claim 56, wherein the at least one clause includes a dummy variable.

58. The method of claim 29, wherein at least one constraint corresponding to initial value constraint of a flip-flop is applied in a lazy manner.

59. The method of claim 58, wherein the at least one lazy constraint is replaced with at least one clause that applies the constraint but does not lead to an immediate implication.

60. The method of claim 59, wherein the at least one clause includes a dummy variable.

61. A method of refining an abstract model comprising:

a) adding interface constraints to a sequential circuit model to capture equality between an input and output of a flip-flop across successive time frames;

b) generating Boolean constraints for a counterexample on the abstract model separately for each time frame up to depth a of the counterexample;

c) assigning k = 1;

d) formulating a SAT problem for checking constraints at depth k;

e) using a SAT solver with resolution-based proof analysis to determine if the SAT problem is satisfiable;

f) setting k = k + 1 if the SAT problem is satisfiable, going to step f if k does not exceed d and stopping if k exceeds d; and

g) refining the abstract model using marked constraints for unsatisfiability at depth k if the problem is not satisfiable.

62. The method of claim 61, wherein at least one external constraint is applied in a lazy manner.

63. The method of claim 62, wherein the external constraint is replaced with at least one clause that applies the at least one constraint but do not lead to an immediate implication.

64. The method of claim 63, wherein the at least one clause includes a dummy variable.

65. The method of claim 62, wherein at least one constraint corresponding to initial value of a flip-flop is applied in a lazy manner.

66. The method of claim 65, wherein the at least one lazy constraint is replaced with at least one clause that applies the constraint but does not lead to an immediate implication.

67. The method of claim 66, wherein the at least one clause includes a dummy variable.

68. A method of verifying correctness of a model of a circuit comprising:

a) obtaining an abstract model by using iterative abstraction;

b) obtaining a correctness property;

c) performing bounded model checking on the abstract model;

d) concluding that a counterexample does not exist at depth k for the circuit if no counterexample exists at depth k for the abstract model; and

e) concluding that the circuit is correct up to depth k if no counterexample exist in step d.

69. A method of performing proof by induction on a circuit comprising:

a) obtaining an abstract model of the circuit by using iterative abstraction;

b) obtaining a safety property;

c) performing proof by induction using bounded model checking approach on the abstract model; and

d) concluding that the proof by induction succeeds on the original circuit if the proof succeeds in step c.

70. A method of performing a symbolic model checking for a circuit using a BDD based approach comprising:

a) obtaining an abstract model of the circuit by using iterative abstraction;

b) generating a correctness property;

c) performing BDD-based symbolic model checking on the abstract model; and

d) concluding that the correctness property is proven true on the circuit if the correctness property is proven true in the abstract model.

71. A method of performing a symbolic model checking of a circuit using a hybrid SAT and BDD based approach comprising:

a) obtaining an abstract model of the circuit by using iterative abstraction;

b) obtaining a correctness property;

c) performing a hybrid SAT and BDD-based symbolic model checking on the abstract model; and

d) concluding that the correctness property is proven true on the circuit if the correctness property is proven true in the abstract model.

72. A method of performing verification of a circuit comprising:

a) obtaining an abstract model of the circuit by using iterative abstraction;

b) performing a BDD-based reachability analysis on the abstract model;

c) adding BDDs corresponding to reachable sets as Boolean constraints in the circuit model;

d) performing bounded model checking proof by induction on the circuit model; and

e) concluding that the circuit is safe if the proof succeeds in step d;

73. A method of performing verification of a circuit comprising:

a) obtaining an abstract model of the circuit by using iterative abstraction;

b) obtaining a correctness property of the circuit;

c) performing a BDD-based reachability analysis on the abstract model;

d) adding BDDs corresponding to reachable sets as Boolean constraints in the circuit model;

e) performing bounded model checking search on the circuit model; and

c) concluding that the circuit is safe up to depth k if no counterexample is found in step e.

74. A method of verifying a safety property for a sequential design, comprising:

a) checking the safety to hold at all depths $i < d$, when starting from the initial state;

b) checking that a negated property is unsatisfiable at depth d, and

c) checking that marked constraints for a proof of unsatisfiability at depth d do not include any initial state constraint for any flip-flops;

d) proving the property to be true for all depths by induction.